Timothy Yu Zhiwen - Project Portfolio for Horo

1. Overview

Tasked to enhance or morph a simple command line interface (CLI) desktop address book application, our team of 5 Computer Science Students decided to morph the program into a calendar scheduling application for students. From the word "Horology", we came up with the name **Horo** for our program. It is capable of maintaining a calendar and to-do list, posting timely reminders for the users.

Here is a screenshot of **Horo**:

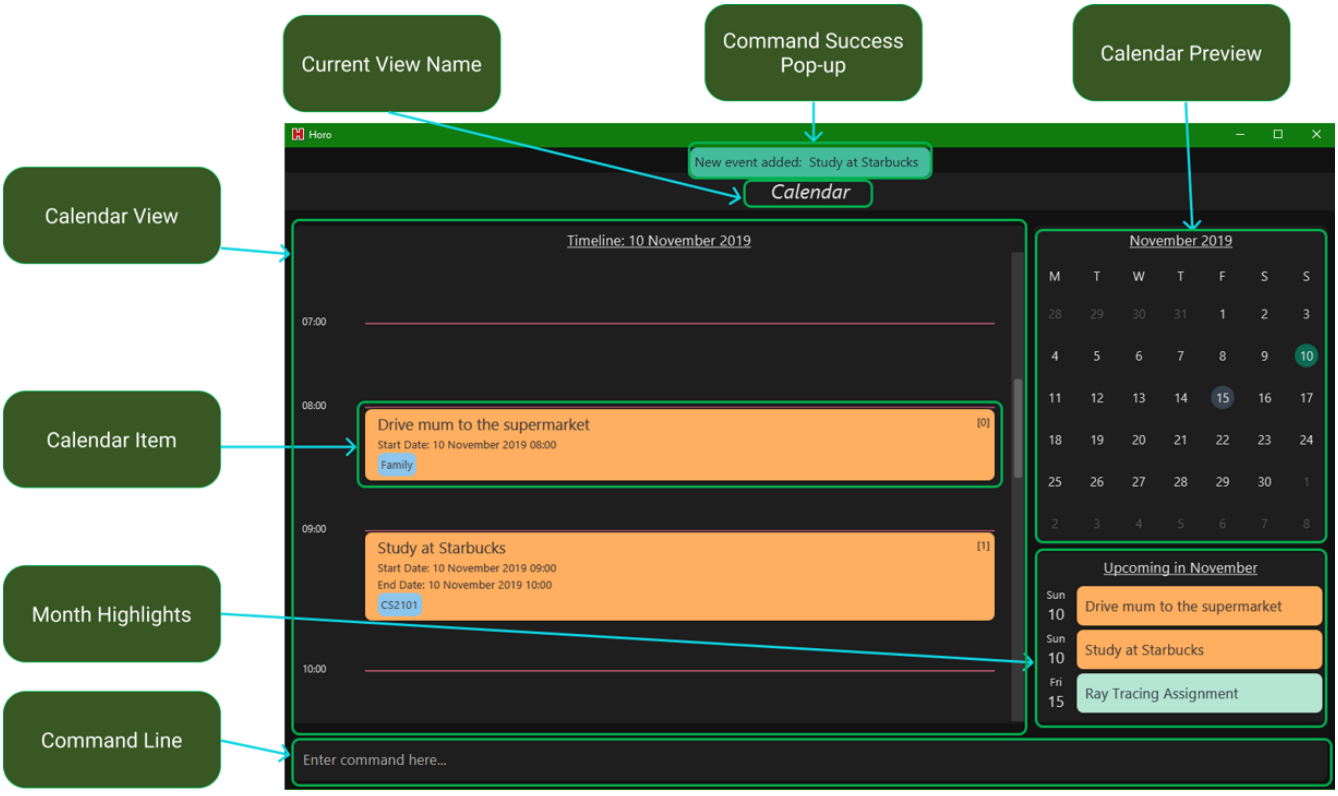

Figure 1. Screenshot of Horo's Graphical User Interface

I played the role of designing the User Interface (UI) and the logic of how my teammates' code will interact with the UI. The following sections below will display the following morphing I did in detail, including certain documentation added to the User and Developer Guide.

Note: The following symbols and format are explained as:

Table 1. Symbol and Formats

`list`	A bold text with highlight as well as the change to monospaced indicates a command input by you which is to be executed by the Horo.
`TimelineView`	A text with highlight as well as monospaced font indicates a component, class or object in the structure of **Horo**

2. Summary of contributions

2.1. Enhancement Ui and Calendar Logic

2.1.1. What is it:

There are 3 different UI screens/panels you can switch between: Calendar Panel, List Panel, Log Panel. Each of the panels shows you a different screen provided with different information.

2.1.2. What does it do:

This is accessed using the commands calendar, list, log respectively and provides a different screen to you depending on what you wish to see.

2.1.3. Why it is necessary:

It provides you a Graphical User Interface (GUI) so you can easily view their Events or Tasks with a simple switch between the views.

2.2. Code Contributed

2.2.1. Sample Code Contributed:

Here is a `Repo Sense Link` that indicates the amount of code I have written for the project. Additionally, our code has an authorship signature, with mine being - `//@@author Kyzure`, or `<!--@@author Kyzure -->`. This authorship can be found at the top of any of the class, or method.

- `Ui Controller Code` - The packages `systemTray` and `listeners` are not written by me, as well as code without my authorship.
- `JavaFX Code for .fxml files`
- `Calendar Date Model Code`

- `Command` `Code` - Only the following is written by me: `DayViewCommand`, `WeekViewCommand`, `MonthViewCommand`, `CalendarViewCommand`, `ListViewCommand`, `LogViewCommand`.

2.3. Other Contribution

2.3.1. Documentation:

- Provided a baseline for both User and Developer Guide at the beginning, and improve the design to make it more readable for the user. (Pull requests #11, #94, #95, #119, #128, #226)

2.3.2. Community:

- Reviewed Pull Requests: #41

3. Contributions to the User Guide

The following sections are an excerpt from my contribution to the User Guide to indicate my ability for writing readable help sections to the user.

3.1. UI

The following commands are related to the changing the display of the UI.

Take note that UI-related commands are not affected by the `undo` and `redo` commands (For more information, see [Undo-Redo]).

3.1.1. Changing Screen View to Calendar View

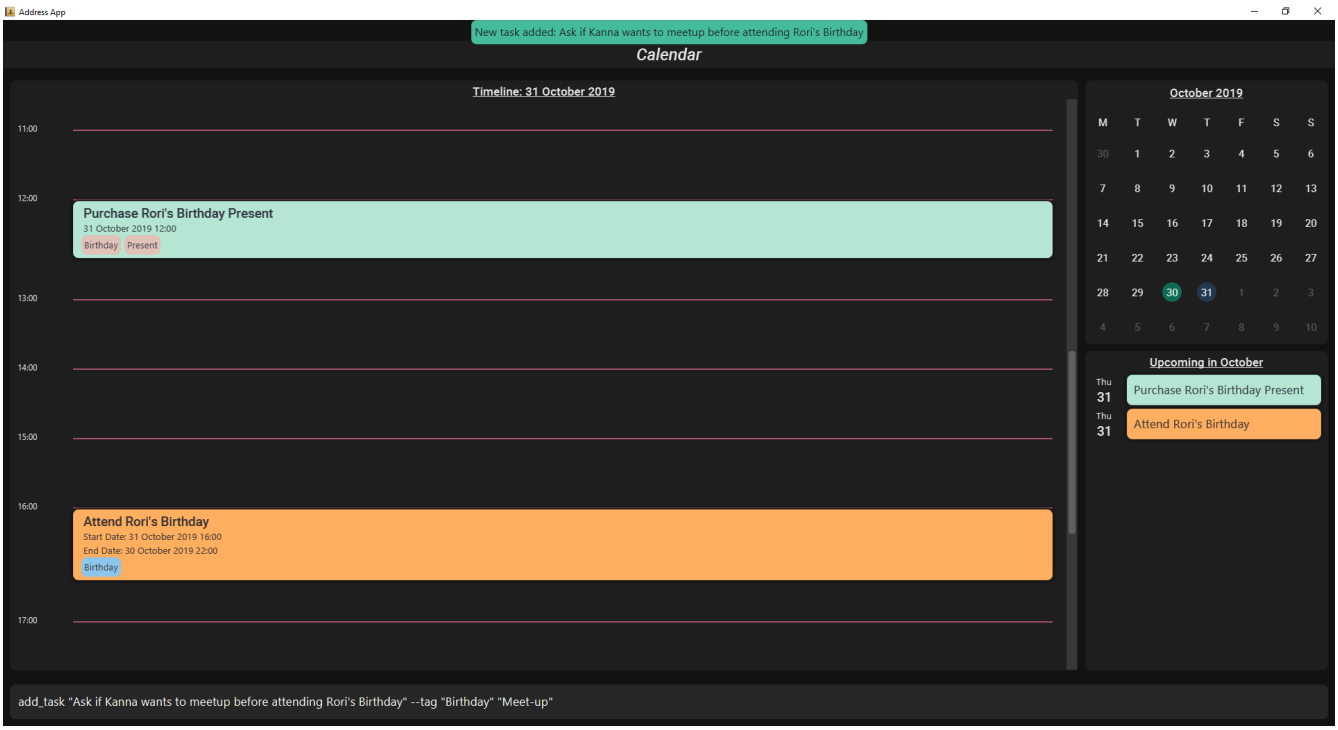

Figure 2. Calendar View Command for Calendar

The `calendar` command switches the display to the Calendar View, which displays a calendar of the specified month and year in addition to a timeline of the specified day, week or month.

The Calendar View will display the specified date. If no date is specified, the last specified date will be displayed. This defaults to the current date.

Upon the initial launch of the application, the timeline and calendar dates will be set to the system's current date.

Command Format:
`calendar`

Command Parameters:
`--date MONTH_YEAR`

Argument Format:
`MONTH_YEAR` : `MM/YYYY`

Example:
`calendar` : Switches back to calendar view without changing the date.
`calendar --date 10/2019`

3.1.2. Changing Timeline to a given day

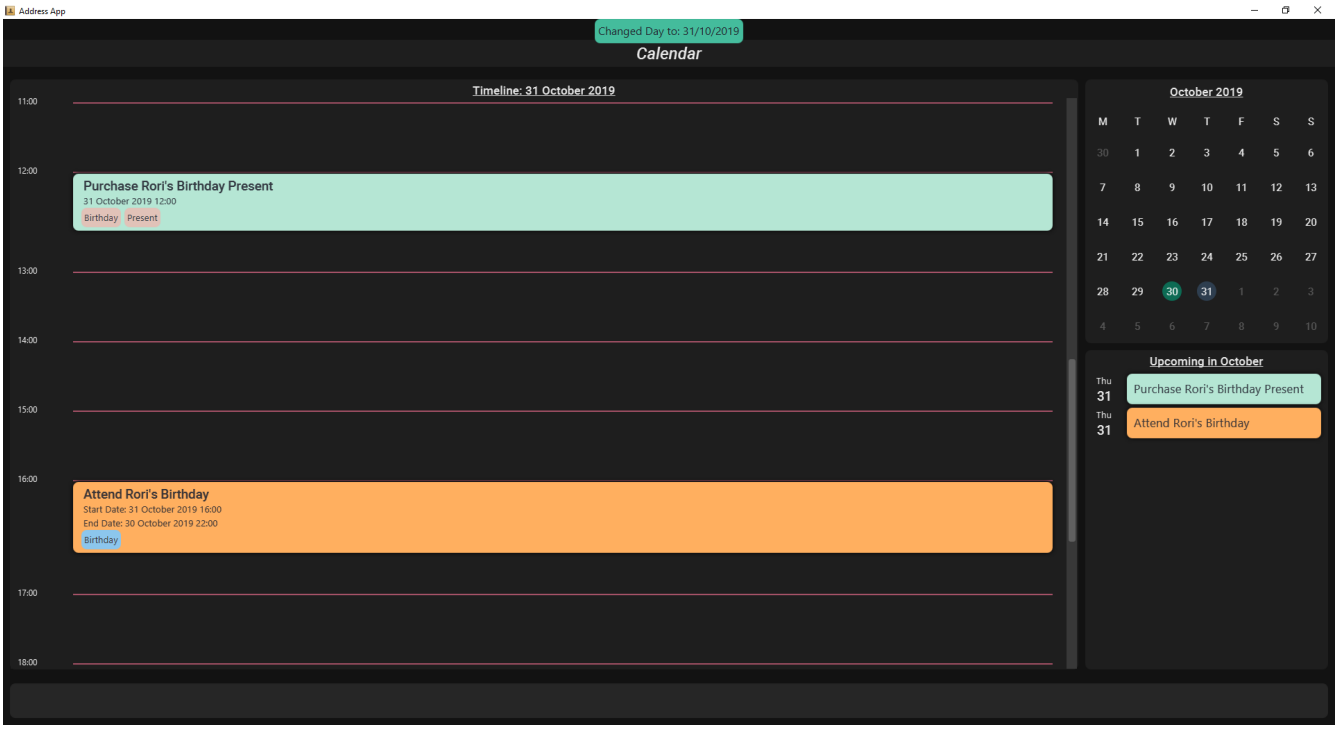

Figure 3. Day View Command for the Timeline

The `day` command sets the timeline in the Calendar View to that of the specified day. Furthermore, this command will switch the current view to Calendar View as well.

Command Format:
`day DATE`

Argument Format:

`DATE : DD/MM/YYYY`

Example:
`day 11/10/2019`

4. Contributions to the Developer Guide

The following sections are an excerpt from my contribution to the Developer Guide to showcase my prowess in the technical aspect of the project.

4.1. UI Component

4.1.1. Implementation during change in Events and Tasks

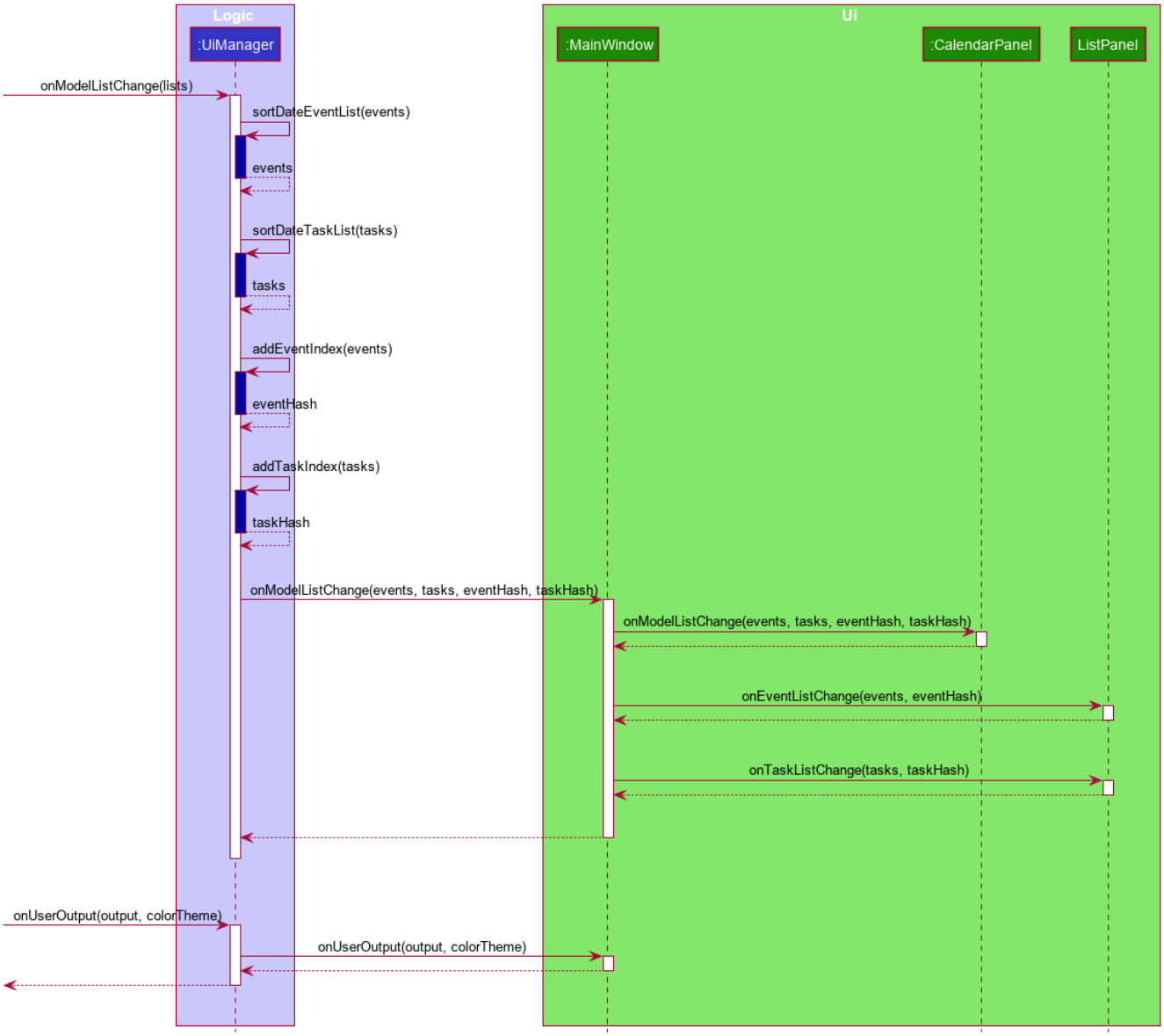

Figure 4. A general Sequence Diagram during a change in the Event and Task Lists model.

The UI system is managed by the `UiManager`, which is found in `Logic` component and is responsible for any change in the models and hence updating the necessary UI portions. The UiManager then

holds a single instance of the `MainWindow`, which represents the base of the UI, and holds the different panels of the UI. Here is the sequence of a change in Events and Tasks for the UI.

Step 1. `UiManager` will be called using `onModelListChange(lists)` method. This will, in turn, take in the `ModelLists`, split them into the `events` and `tasks`, and sort them. Afterward, two HashMaps, `eventHash` and `taskHash` are created to deal with the indexing of the UI later on.

Step 2. `MainWindow` will be called by `UiManager` using `onModelListChange(events, tasks, eventHash, taskHash)`, which will in turn proceed to call the methods that will update the different views represented by:

- `CalendarPanel` - `onModelListChange(events, tasks, eventHash, taskHash)`
- `ListPanel` - `onEventListChange(events, eventHash)` and `onTaskListChange(tasks, taskHash)`

Step 3. `UiManager` will also be called using `onUserOutput(output, colorTheme)`, which will in turn call `onUserOutput(output, colorTheme)` for MainWindow.

As for these 3 main panels, each of them will be explained further below

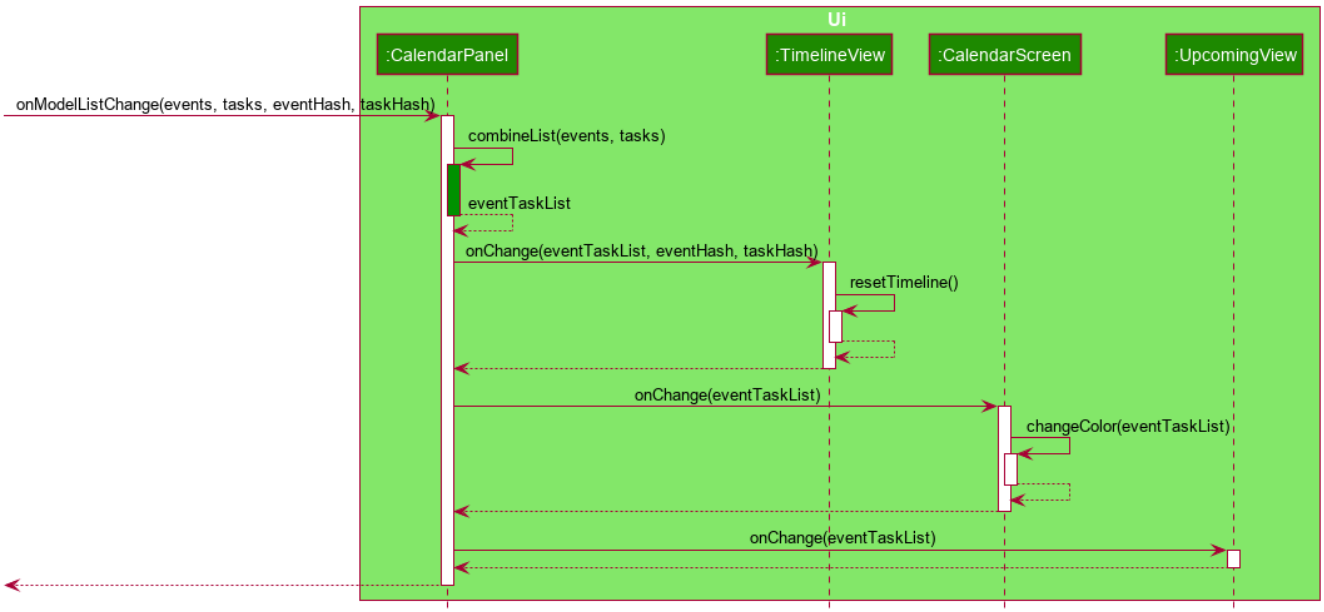

Figure 5. Sequence Diagram for `CalendarPanel`

Step 2.1. `CalendarPanel` will be called by `onModelListChange(events, tasks, eventHash, taskHash)`, and will proceed to zip the two lists into a single list for sorting purposes.

Step 2.2. Afterward, it will call `onChange` for the 3 smaller components:

- `TimelineView` - When called, it will reset the current timeline using `resetTimeline()`
- `CalendarScreen` - When called, it will change the calendar to the given date, as well as calling `changeColor(eventTaskList)` to change the color of a day in the calendar.
- `UpcomingView` - When called, it will simply reset the view to input the correct events and tasks.

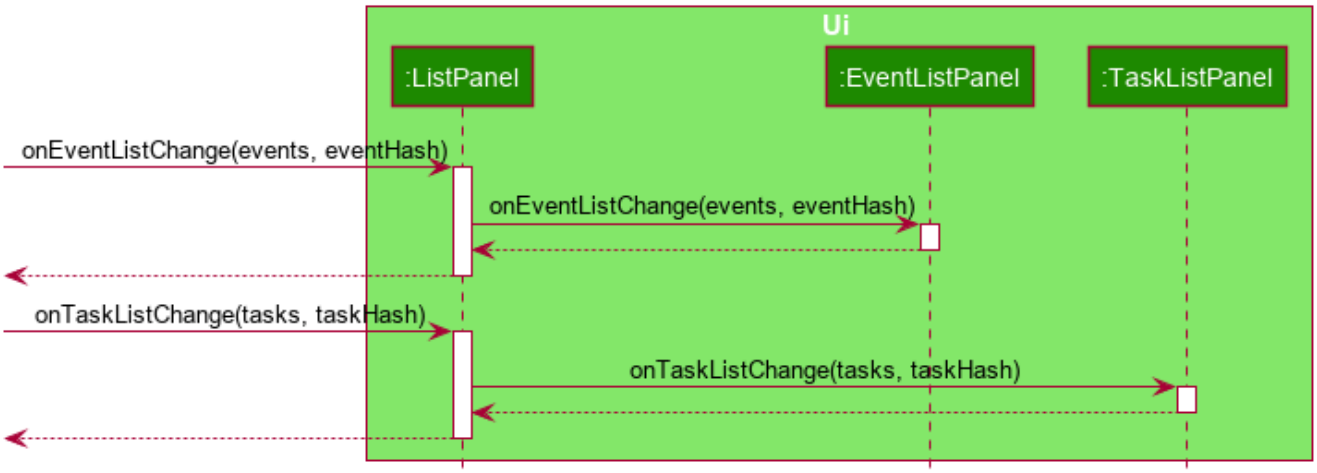

Figure 6. Sequence Diagram for `ListPanel`

Step 2.3. `ListPanel` will be called using `onEventListChange(events, eventHash)` first. It will proceed to call `EventListPanel` to change the list according to the given list of events.

Step 2.4. Additionally, `ListPanel` will also be called using `onTaskListChange(tasks, taskHash)`, which will eventually call `TaskListPanel` to change the list accordingly as well.

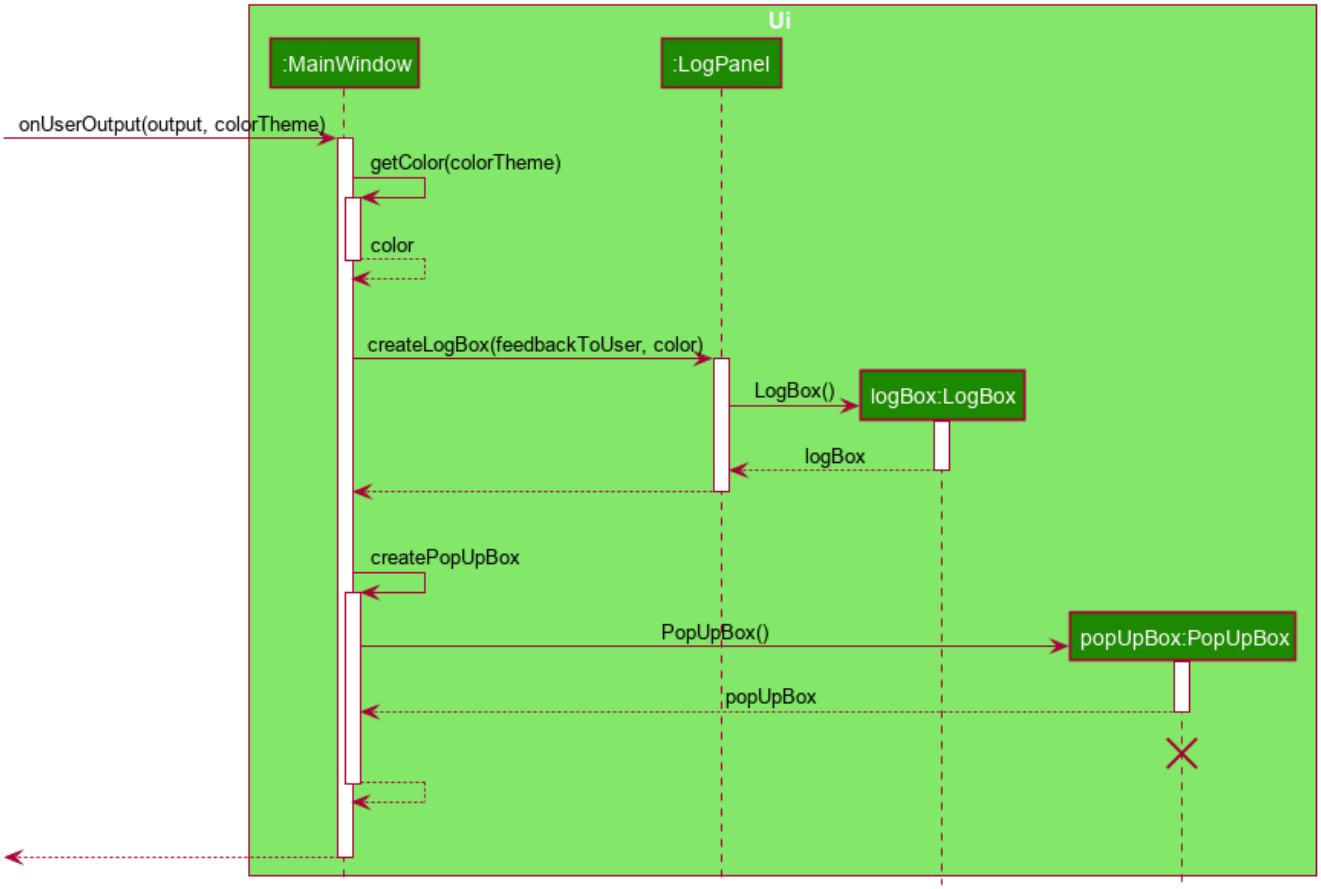

Figure 7. Sequence Diagram for `LogPanel`

Step 3.1. When `MainWindow` gets called using `onUserOutput(output, colorTheme)`, it will proceed to get the actual color scheme in the form of a `String`, and creates 2 different boxes to display the output.

Step 3.2. It will call `LogPanel` to create a `LogBox` using `createLogBox(feedbackToUser, color)` to display the output to the user in `LogPanel`

Step 3.3. Next, it creates `PopUpBox` and display it temporarily on any of the panels, and proceed to unused afterward.

4.1.2. Implementation when changing the date of timeline

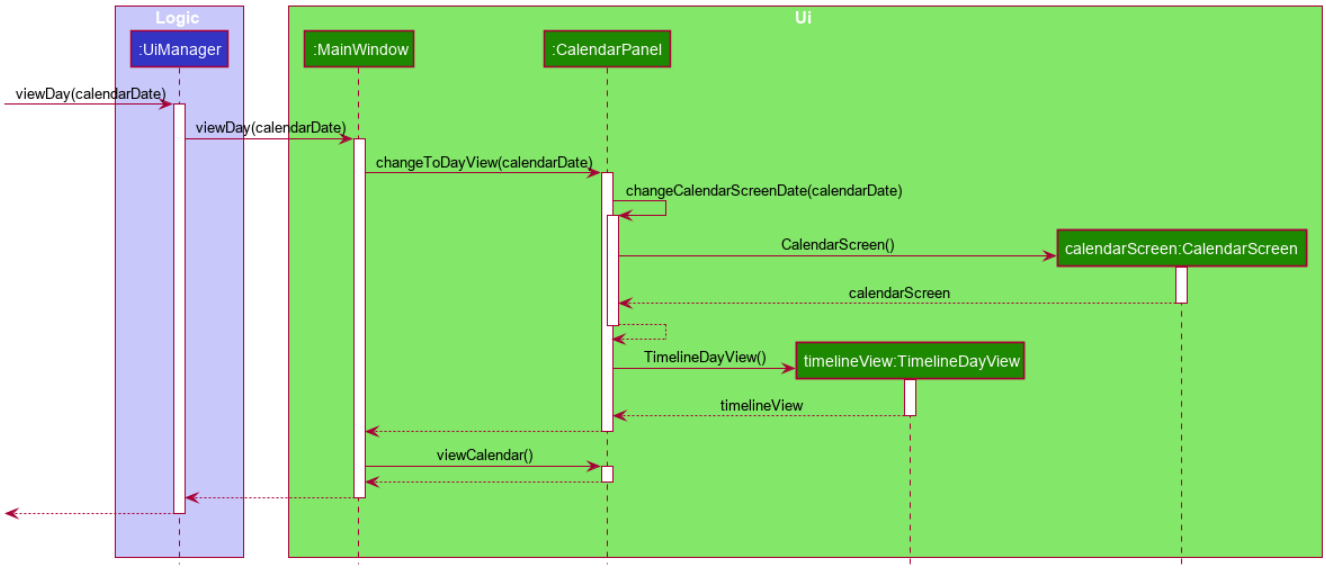

Figure 8. Sequence Diagram for changing the timeline date

Here is an example of the sequence for the UI when `DayViewCommand` is executed to change the date of the timeline.

Step 1. When the command is executed, it will proceed to call `UiManager` through `viewDay(calendarDate)`, which in turn will call `MainWindow` and subsequently `CalendarPanel`.

Step 2. `CalendarPanel` will proceed to execute `changeCalendarScreenDate(calendarDate)`, which will create an instance of `CalendarScreen` to display the calendar.

Step 3. Afterward, a new instance of `TimelineDayView` will be created to display the timeline.

Step 4. Lastly, `MainWindow` will call `viewCalendar` which will be explained in the next section, allowing `CalendarPanel` to be visible while the other panels remain invisible.

4.1.3. Design Considerations

The design considerations are more towards how the appearance of the UI, as well as how the architecture of the code would have changed depending on such appearance.

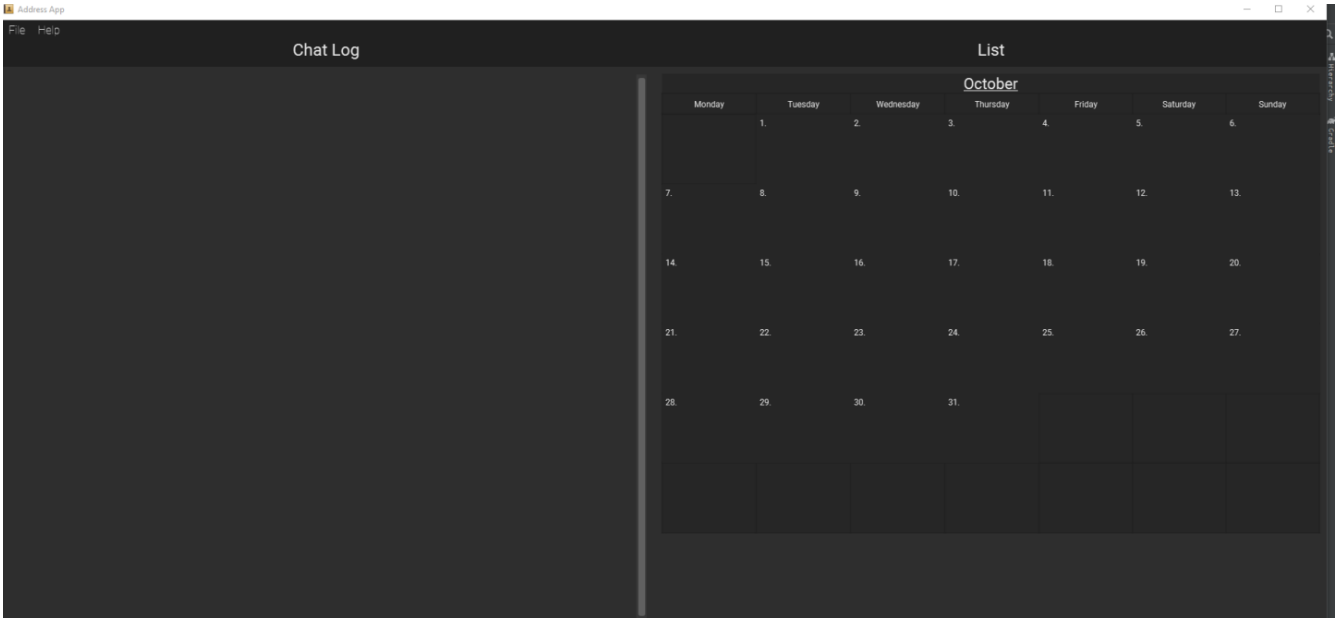

Figure 9. Old design of the UI

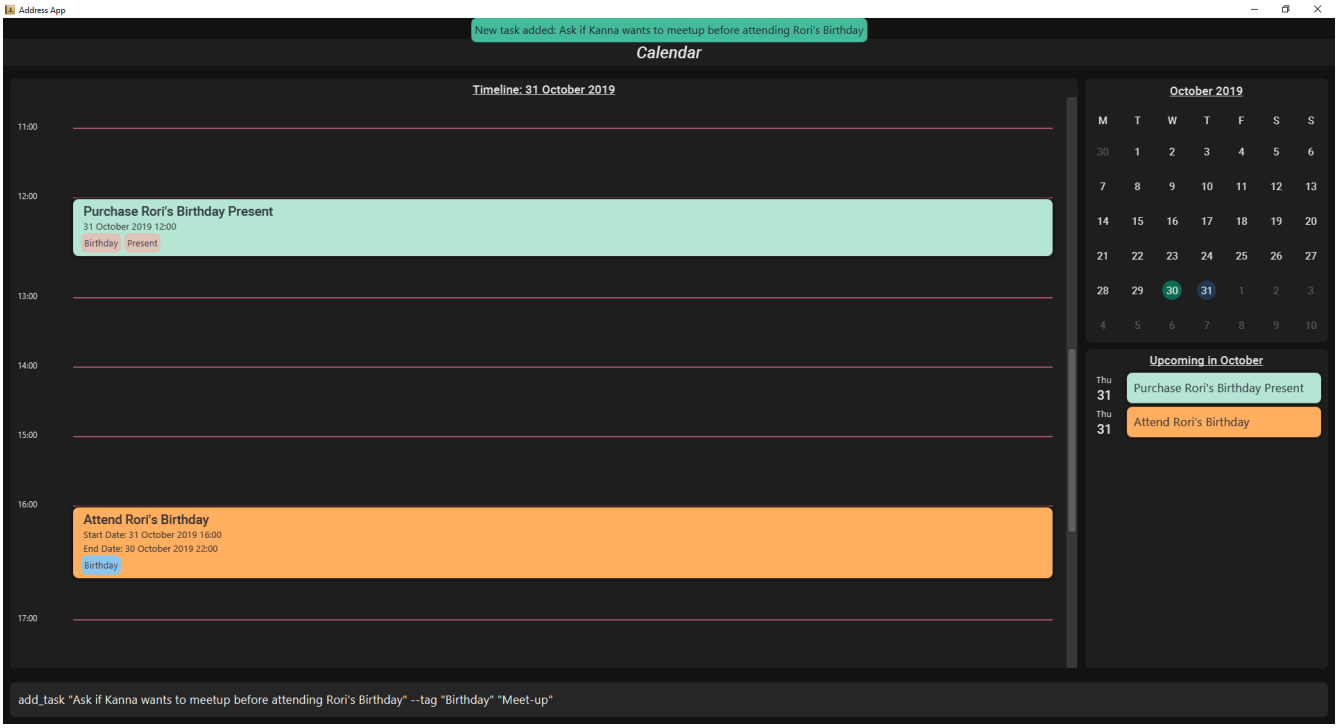

Figure 10. Current design of the UI

Aspect: Design of the `CalendarPanel`

- Alternative 1: The `CalendarPanel` is of an actual calendar, depicting a limited number of events and tasks on each day of the month.
    - **Pros**: It will provide a better representation of a calendar, allowing people to judge how much is going on in a day of that month in one look.
    - **Cons**: Due to the nature of how limited in size a calendar can be, the user will be required to either check `ListPanel` for the details of an event or task, or have an extra screen beside the calendar for the user to check the details.
    - **Cons**: Similarly, a calendar can only input up to a fixed amount of events or tasks there are on a particular day.

- Alternative 2 (current choice): The `CalendarPanel` consists of a mini-calendar as well as a timeline. An additional slot for upcoming events and tasks was later designed with an increase in space.
    - **Pros**: Provides a much greater space to show how much events or tasks one can have in a day, week or month.
    - **Pros**: The user can easily manage and check the Events and Tasks of a certain day.
    - **Cons**: Even though it is a timeline, it is still rather similar to list view, just with the timeline added to limit the number of events or tasks seen on that day, week or month.
    - **Cons**: The user will not be able to easily know what Events or Tasks there are, unless they change the view to Month view. On the other hand, the increase in space allows a small section for the upcoming events and tasks which tackles this problem.

Aspect: Design of the `LogPanel`

- Alternative 1: The `LogPanel` is placed side-by-side with any other panel.
    - **Pros**: The users can always have a visualization of the success of their commands
    - **Cons**: A large portion of the space is used for the `LogPanel`, even if it is scaled down compared to the other panels.
    - **Cons**: Appearance-wise, it looks extremely clunky due to most of the users' time will be looking at the calendar or list itself instead of the log.
- Alternative 2 (current choice): The `LogPanel` is placed separately as a different panel that can be accessed at any time from other panels. After each command is typed, a pop-up box will appear to indicate the success or failure of the command.
    - **Pros**: Most of the time, users would only want to know if their command is successful or not. Thus having the pop-up box will be sufficient for such an indication.
    - **Cons**: The user will have to check the `LogPanel`

The initial design is as of the image above showing the old UI. However, we decided to scrape it and did an overhaul of the UI using alternative 2 instead. This is due to our decision of wanting a better-looking and minimalist UI instead of one packed with information.